Development of site-oriented Analytics for Grid computing centres

2015 J. Phys.: Conf. Ser. 664 022041

(http://iopscience.iop.org/1742-6596/664/2/022041)

Journal of Physics: Conference Series **664** (2015) 022041 doi:10.1088/1742-6596/664/2/022041

Development of site-oriented Analytics for Grid computing centres

A. Washbrook[2], D. Crooks[1], G. Roy[1], S. Skipsey[1], G. Qin[1], G. P. Stewart[1], D. Britton[1]

[1] Kelvin Building, Physics and Astronomy, University of Glasgow, Glasgow, G12 8QQ
[2] SUPA, School of Physics and Astronomy, The University of Edinburgh, James Clerk Maxwell Building, Mayfield Road, Edinburgh, EH9 3JZ, United Kingdom

E-mail: awashbro@ph.ed.ac.uk, david.crooks@glasgow.ac.uk

Abstract. The field of analytics, the process of analysing data to visualise meaningful patterns and trends, has become increasingly important in scientific computing as the volume and variety of data available to process has significantly increased. There is now ongoing work in the High Energy Physics (HEP) community in this area, for example in the augmentation of systems management at WLCG computing sites. We report on work evaluating the feasibility of distributed site-oriented analytics using the Elasticsearch, Logstash and Kibana software stack and demonstrate functionality by the application of two workflows that give greater insight into site operations.

1. Introduction

Each computing site pledging resources to the Worldwide LHC Computing Grid (WLCG) generates a wealth of monitoring data from numerous sources as part of their ongoing operations. These include system logging, Grid middleware services, network and storage utilisation, local resource management scheduling information and user application performance. This rich dataset can be consolidated and exploited by the deployment of emerging analytics platforms to enable post-facto diagnostics to give a more comprehensive understanding of site system activities.

In this study we evaluate the components necessary to enable distributed site-oriented analytics at two computing sites in the NGI UK distributed Tier-2 ScotGrid facility. A site-oriented platform allows site administrators to more easily leverage their logging and monitoring data to determine underlying trends in workload performance and system health that may be unclear from a disparate data sources. A distributed platform provides wider analytics coverage for error diagnosis and the identification of patterns in workflow common to multiple sites that is not apparent from an individual site perspective.

The approaches taken to categorise, transport and store relevant datasets will be discussed with a focus on the choice of analytics tools available. The applications on such a platform will then be explored by the use of two examples that provide insight into Grid middleware and job management workflows across Grid computing centres.

2. Approach

For this work we require a set of tools that can aggregate and curate logfiles, generate events based on these with rich metadata, store the resultant events in a searchable manner and (optionally) provide a visual search interface.

The Elasticsearch, Logstash and Kibana (ELK) stack [1] is a combined analytics, logging and visualisation platform that has risen in prominence in the HEP community. Logstash in this context performs the log aggregation, while Elasticsearch stores events generated from these logs and Kibana provides a visual interface to searching Elasticsearch data. Unstructured data from numerous sources can therefore be gathered, curated and then visualised through a single extensible interface. In our approach we combine data collected from ELK stack instances located at the two participating Grid sites into a prototype regional ELK instance.

The approach we took in this work originated from an initial examination of the underlying processes in an ELK stack. In particular, Logstash functions by using an agent on each host which gathers data and passes it by some medium to a Logstash server. The data is then parsed and Elasticsearch ingests the events that are subsequently generated. We considered two approaches - firstly a distributed Elasticsearch which covered both sites into which local Logstash instances sent data, and a second approach where each site Logstash sent data to an additional, regional Logstash instance which aggregated the data and passed it to a regional Elasticsearch instance.

We felt that the latter approach, wherein the site Logstash instances forwarded data to a regional Logstash instance, was more flexible, as it meant that the data which was aggregated could be parsed twice; once reflecting local policies, and again by the regional Logstash. This maximised the potential to tune the data ingested.

In particular, this approach would allow for checks at each level to ensure that sensitive data was cleaned appropriately.

2.1. Transport

There are a number of different techniques that can be used to transport Logstash data from a Logstash agent to a Logstash collector. In this work, for the purposes of simplicity and clarity, we focussed on using the lumberjack protocol [2] which uses a standard TLS configuration to secure data transport. To standardise our approach, we used *logstash-forwarder* [2], an implementation of the Logstash agent, for the host agents. In each of the two sites we deployed a Logstash agent which performed initial parsing of the log data and then passed it both to a local Elasticsearch instance and also to the regional Logstash collector (see Figure 1). For the purposes of this exercise the regional collector was based at the Glasgow site.

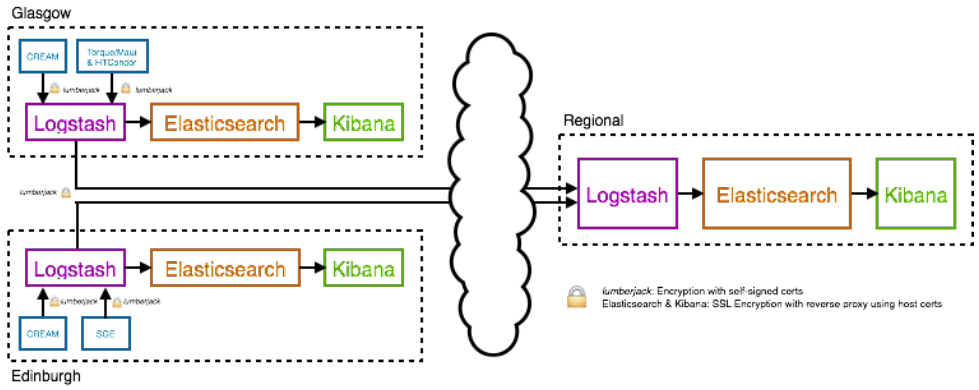

Figure 1. Process flow from hosts to regional Logstash instance

2.2. Security Considerations
A necessary component of a distributed platform approach is the transfer, storage and examination of potentially sensitive information outside of a site domain. It was therefore important to consider security and privacy measures from the outset. In this study the following security measures were implemented:

- Each Kibana instance (and Elasticsearch through a reverse proxy) was available only through HTTPS with access restricted to explicitly listed valid grid credentials
- Potentially sensitive user and location data forwarded by Logstash-forwarder instances - including local account details, delegation ID mappings and IP addresses - were removed prior to transport off the site domain by stripped or anonymising identified strings as data was ingested into Logstash
- Data transported to each Logstash service was sent encrypted using the lumberjack protocol, using a standard TLS infrastructure using self-signed host certificates
- Host-level firewalls were configured to restrict access to ports used by Elasticsearch and Logstash

3. Applications
Once the data was harmonised and stored in site and regional Elasticsearch instances it was possible to apply detailed analytics to gain further insight into site operations that would be less trivial to construct from unstructured multiple data sources. The potential of such a platform is shown by two example applications built using the model described in the Section 2.

3.1. CREAM Computing Element Log Data
The logs for the CREAM Computing Element (CE) middleware service [3] were chosen for one application as it was a common service used on both sites. The test case was focussed on harvesting the state changes of jobs, with the idea that this could be useful in looking for correlations between the two sites in the types of state changes being seen. Moreover, we restricted the logs gathered to one particular CREAM logger which meant that we were dealing with a simple case where the information gathered could be controlled in a relatively straightforward fashion. To understand the workflow, the individual stages of the process are discussed below. In each case only the key logstash actions are discussed.

3.1.1. logstash-forwarder This service has a simple configuration, selecting one logfile and tagging it as a cream log.

```
"paths": [ "/var/log/cream/glite-ce-cream.log" ],
"fields": { "type": "cream" }
```

3.1.2. Site logstash The configuration of the site logstash has a configuration that firstly selects out cream logs, and then the specific logger used in this test. For this work, a key section is the following configuration where the `replace` action is taken: this operation adds the extra information required for the regional logstash to parse the site logstash data correctly.

```
mutate { replace => [ "message", "SITE_LOGSTASH cream %{message}" ] }
```

The `match` action below was used to parse the cream log for our particular logger `AbstractJobExecutor` wherein custom patterns had been created for clarity:

```
match => [ "cream_message" , "JOB %{CREAM_ID:cream_id} STATUS CHANGED:
           %{CREAM_STATUS:cream_initial_status} => %{CREAM_STATUS:cream_final_status}
            %{GREEDYDATA:rest_of_message}" ]
```

Finally, a `mutate` action was used where the user information are redacted, including

```
gsub => [ 'cream_message' , 'USER_ID=".*"; ' , 'USER_ID=REMOVED; ' ]
```

The output of this process was then directed both to a local Elasticsearch instance and to the regional Logtash collector.

3.1.3. Regional Logstash The configuration of the regional Logstash is very similar to that of the site Logstash, but the `match` action looks for the extra data included as part of the site Logstash parse step.

```
match => [ "message" , "%{TIMESTAMP_ISO8601:site_timestamp}
            %{HOST:origin_host} %{HOST:site_logstash} %{WORD:type}
             %{CREAM_TIMESTAMP:cream_timestamp} %{WORD:cream_priority}
              %{NOTSPACE:cream_logger} - %{GREEDYDATA:cream_message}"]
```

A second match step is then used to match the specific logger data in the same way as in the site logstash case.

3.1.4. Results Following the harvesting of this data, the results were visualised using Kibana; Figure 2 shows one representation of this output.

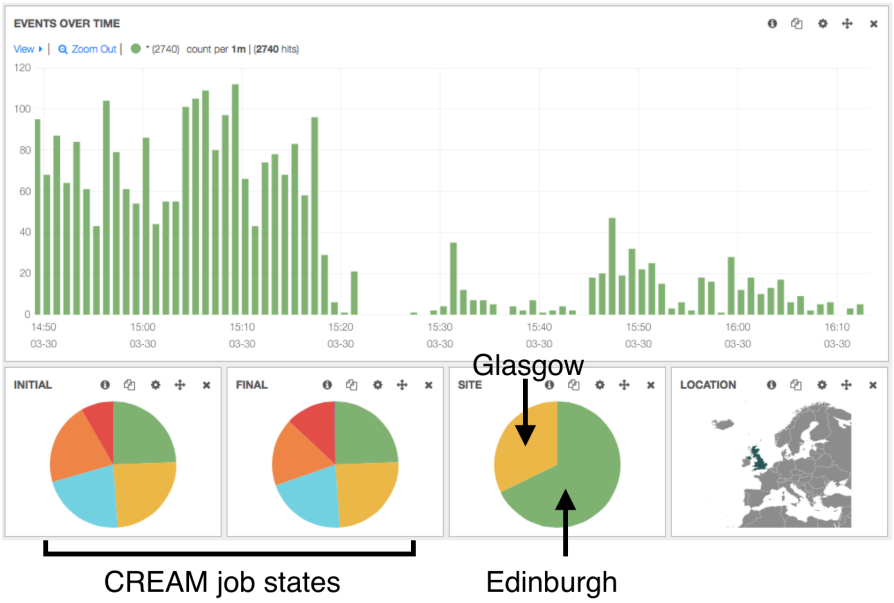

Figure 2. Sample Kibana visualisation of job state events drawn from two Grid computing sites

We now move on to discuss the second application which changes the focus to local resource management system analytics.

3.2. Gridengine Job Accounting

This example demonstrated how useful time-series data can be derived from event-based data stored in Elasticsearch. In particular it is shown how trending data in Grid job queuing times can be extracted from local resource management system (LRMS) job accounting records.

Logstash was configured to parse Gridengine [4] accounting record file entries for each job completed on the Edinburgh site compute resources. Each field in a job accounting record entry was converted into an appropriate type (e.g. string, integer or float) to standardise further manipulation of the data. Similarly, date fields were converted to a consistent format to allow time-based search requests through the Elasticsearch API [5] and the Kibana user interface.

Additional job metadata was derived from the accounting record fields as part of the Logstash ingestion process and stored in Elasticsearch in step with the original data source to avoid unnecessary recalculation later in the application workflow. In this example additional fields were defined to store simple scheduling information (such as job queueing time) derived from timestamps stored in the job accounting record.

It was then possible to retrieve continually updated queue data for an analytics workflow from a third-party service that periodically fetched results through the Elasticsearch API. An example service was developed that called the Python Elasticsearch API module [6] to extract average queue times over over the previous 4, 12 and 48 hours with the following conditions:

- All jobs run on the cluster that were submitted through a Grid Computing Element
- Any jobs requesting single core resources
- Any jobs requesting multiple core resources

The above requests were also made for each Virtual Organisation (VO) with access to site resources.

Figure 3. Process workflow for the derivation of metrics and their aggregation for use in time series visualisation

The workflow of this application type is shown in Figure 3. In this example the service was run every 10 minutes with average queue time output sent to Graphite [7] for storage. It was decided that Graphite was the best option for storing derived time series data as opposed to Elasticsearch which was more suited to the storage of event-based datasets (e.g. new log entries). Additionally, the adoption of Graphite allowed the Grafana [8] tool to be used to visualise time-series data in a similar manner to the presentation of event-based results in the Kibana interface.

A snapshot of a scheduling analytics prototype dashboard built in Grafana is shown in Figure 4. The rolling averages of overall job queue times for the three chosen sampling periods are shown on the top plot and indicate the transient and long term underlying trends in scheduling conditions. The average queue time per VO shown in the centre plot is useful in tracking relative workload throughput performance for sites providing resources to multiple VOs and (in the case of the Edinburgh site) non-Grid projects. Similarly the overall average queue time trends for jobs requesting different CPU resources (bottom plot) was useful in evaluating scheduling performance for increased demand in multiple-core workloads from LHC experiments.

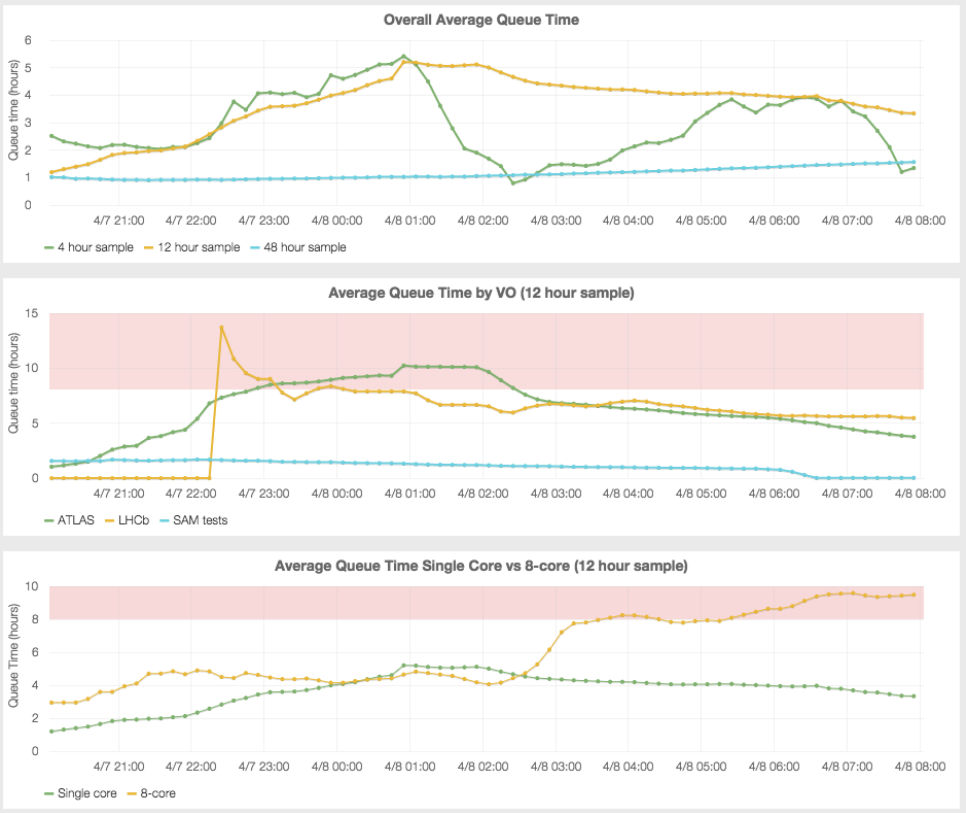

Figure 4. Rolling average queue times displayed by Grafana for all Grid jobs, per VO and for single-core vs 8-core performance

As the data used to create scheduling analytics is retrieved through the Elasticsearch API and not at the data source it was therefore possible to reuse the same application for LRMS implementations at other computing sites with only changes to the Logstash configuration needed. In addition, the same application can be used on consolidated data on the regional instance.

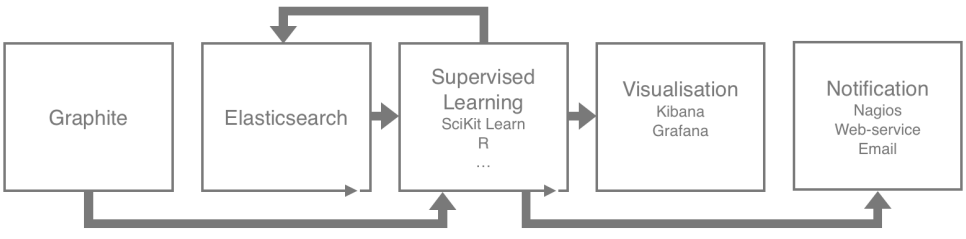

Figure 5. Process workflow for further analytics applications

4. Conclusions

It has been shown that site-oriented analytics for WLCG computing sites is feasible by the use of emerging open-source analytics tools such as Elasticsearch, Logstash and Kibana. Minimal extensions were then required to construct a two tier solution which enabled relevant data to be

forwarded to a regional instance to allow common trends to be identified at the two participating Grid computing sites. Further data analytics were developed by the use of third party tools and customised scripts acting on the primary or derived datasets stored in Elasticsearch. Periodic running of these scripts generated time-series information which captured underlying trends in the site data.

Development in this study was focused on providing a solution for Grid computing sites rather than an attempt to cover all data sources generated by a Virtual Organisation (VO). Indeed, such an approach was intended to complement analytic development from larger VOs (such as the LHC experiments) whilst benefiting smaller VOs who may not have the resources available to develop similar solutions.

Site-specific data collection can be expanded by the deployment of *logstash-forwarder* onto hosts running services of interest and the development of filtering and parsing logic for each new data source in Logstash. Potential data sources relevant to Grid computing include storage interface transaction logs and per-job system resource utilisation on worker nodes.

As site coverage becomes more comprehensive more elaborate analytics applications could be used to trigger automatic notification of error conditions to other monitoring frameworks. A possible workflow of this approach is shown in Figure 5. In the example described in Section 3, it can be envisaged that the queue trends reaching the upper threshold illustrated in the time-series plots in Figure 4 could trigger a notification action. Furthermore future development can be easily extended to harness supervised machine learning techniques to automatically identify emerging trends and spurious activities by using the curated data as a continuous training set. As these techniques become more established issue pre-emption could then be introduced to assist with site operations (e.g. the removal of a badly performing Grid worker node from active service).

Before this development is moved forward best practices on data storage and resiliency need to be considered as well as determining the scaling limitations of data ingestion and transport. As discussed in Section 2, extensive treatment of the security implications would have to be continually addressed with a conservative approach to sharing data between sites. We will look towards developing this work in the context of heightened interest in this area in the HEP and WLCG communities.

References

[1] *Elastic* [online] Available http://www.elastic.co [accessed 15/05/2015]
[2] *Logstash-forwarder* [online] Available https://github.com/elastic/logstash-forwarder [accessed 15/05/2015]
[3] *Computing Resource Execution And Management Service (CREAM)* [online] Available https://wiki.italiangrid.it/CREAM [accessed 15/05/2015]
[4] *Son of Grid Engine Project* [online] Available https://arc.liv.ac.uk/trac/SGE [accessed 15/05/2015]
[5] *Elasticsearch API* [online] Available https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html [accessed 15/05/2015]
[6] *Elasticsearch API* [online] Available https://elasticsearch-py.readthedocs.org/en/master/ [accessed 15/05/2015]
[7] *Graphite* [online] Available http://graphite.readthedocs.org [accessed 15/05/2015]
[8] *Grafana* [online] Available http://ww.grafana.org [accessed 15/05/2015]